Prince Sultan University
College of Computer and Information Sciences

جامعة الأمير سلطان
كلية علوم الحاسب والمعلومات

Major2 Examination

Term: 162 Year: 2016/2017

Course instructor(s): Dr. Sarab AlMuhaideb and Dr. Basit Qureshi

Course title : Data Structures & Algorithms **Course code: CS210**

Exam duration: 50 Minutes **Exam date: 26/04/2017**

Number of exam pages: (6) pages

(Including cover page)

Exam Rules

- Students are **NOT** allowed to **TALK** to each other **DURING THE EXAM**.
- Students are **NOT** allowed to bring **CELL PHONES, BAGS, SUNGLASSES**, and **any ELECTRONIC DEVICES** into the examination hall.
- Students must **SIGN** the PSU Honor Code.
- This exam is closed books and notes.
- Students have to hand in this exam when time is called. Failure to do so would result in 20% penalty.

The PSU Honor Code

أتعهد أنا الطالب الموقع أدناه بأن أكون ملتزماً بالأمانة ومطبقاً لجميع تعليمات وأنظمة الامتحانات في الجامعة

"I do hereby pledge to conduct myself with honesty and integrity during the examination and abide by the rules and regulations of the University."

Don't forget to write and sign the Honor Code pledge!

SOLUTION

Student's Signature:_____

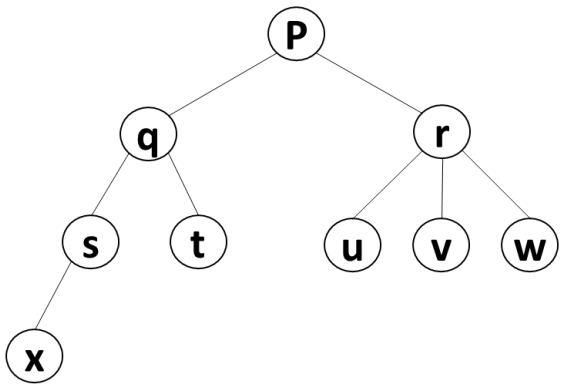
Student's Name: _____

Student's ID#: **Student's Section#**:

Question No.	Student's Score
Part 1-CLO1 / 5	
Part 2-CLO2 / 4	
Part 3-CLO3 / 6	
Total / 15	

Part 1: CLO1 [/ 5]

1. [2] Choose the correct answer based on the shown tree.



1. Which of the following must be done to ensure that this tree is a min-heap?

(a) Remove **x**
(b) Remove one of **u**, **v**, or **w**
(c) Correct the heap-order property.
(d) None of the above

2. Assume that all necessary changes have been made to ensure this tree is a min-heap. After one call to **removeMin**(), which node will be at the root?

(a) **p**
(b) **q**
(c) **x**
(d) **r**

3. FIFO Queues require almost no knowledge about the data they store other than its size. What knowledge (other than size) do priority queues require?

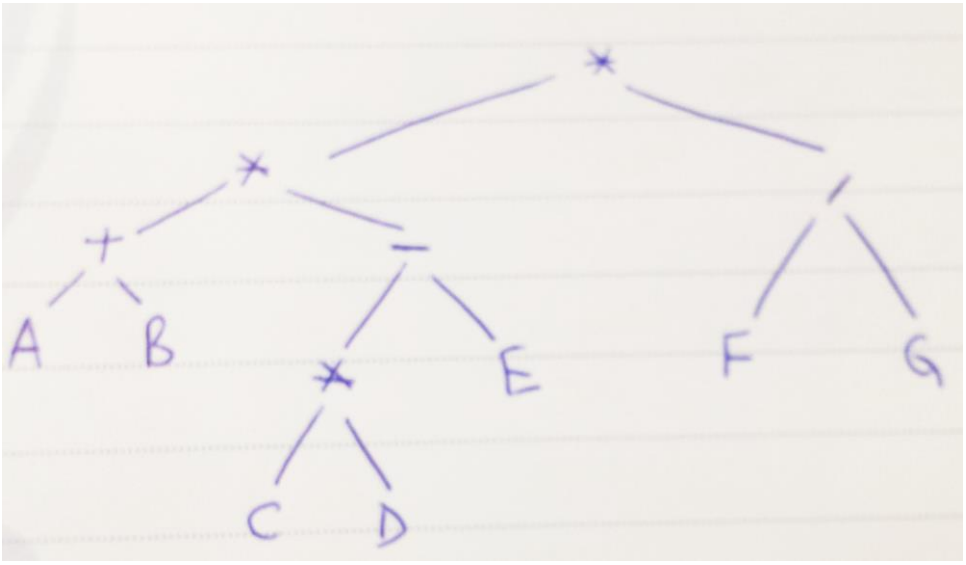
(a) None.
(b) They need to be able to store the priority of the data items.
(c) They need to generate numerical values describing the priority of the data items.
(d) They need to know the exact structure of the data items.

4. Here's an algorithm to sort *n* numbers with a heap. Start with an empty heap and insert each number into the heap sequentially. Then, remove a number from the heap and output it, repeating until the heap is empty again. How long does this take? (Tight upper bound, please.)

(a) $O(\log^2 n)$
(b) $O(n^2)$
(c) $O(n)$
(d) $O(n \log n)$
(e) $O(n^2 \log n)$

2. **[2]** Draw the **expression tree** representing the expression below, then use it to find the **postfix form** for this expression:
   (A + B)∗(C∗D − E)∗F / G

   Expression tree:



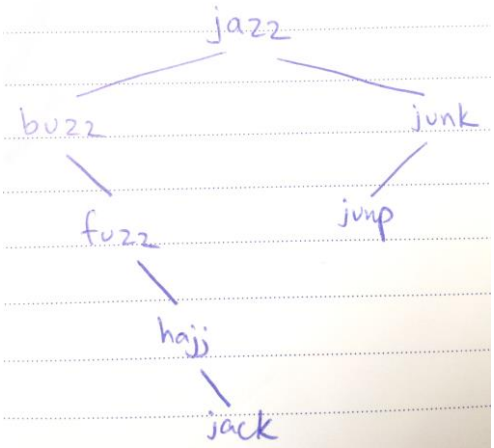
   Postfix form of expression:
   AB+CD*E-*FG/*

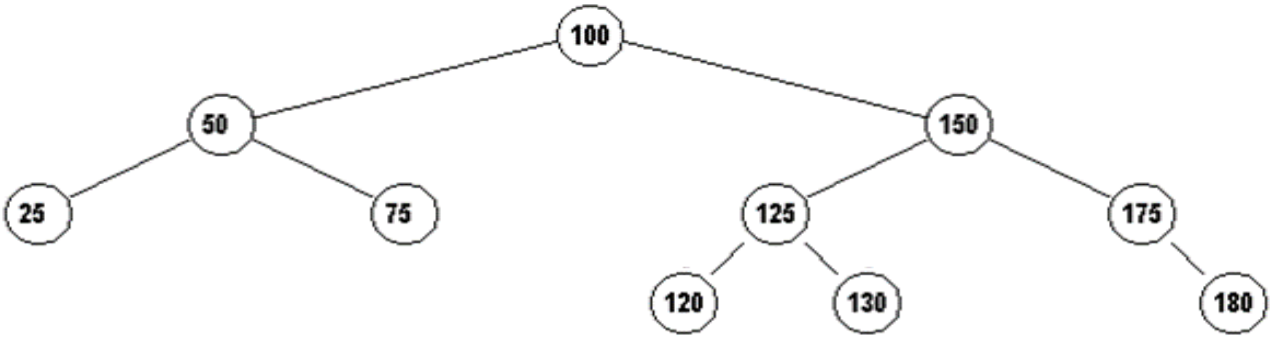
3. **[1]** Assume you have a binary search tree with 30 nodes. In the worst case, what is *maximum possible depth* of any leaf node in this tree?
   30

Part 2: CLO2 [/ 4]

4. **[1]** Draw **a binary search tree of strings** with the following keys

   jazz buzz fuzz hajj jack junk jump



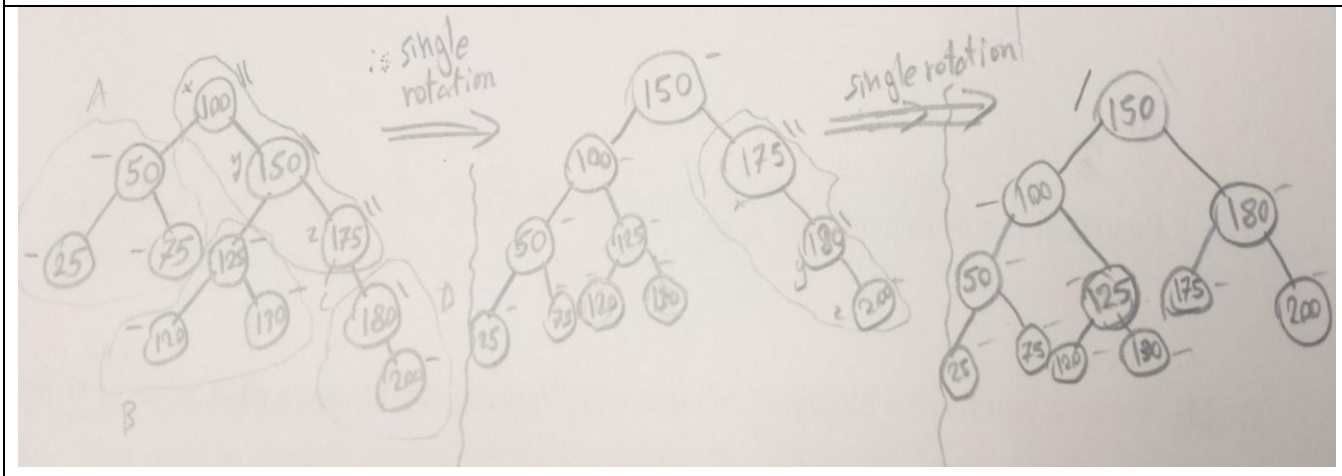


[3] Consider the **AVL tree** below.

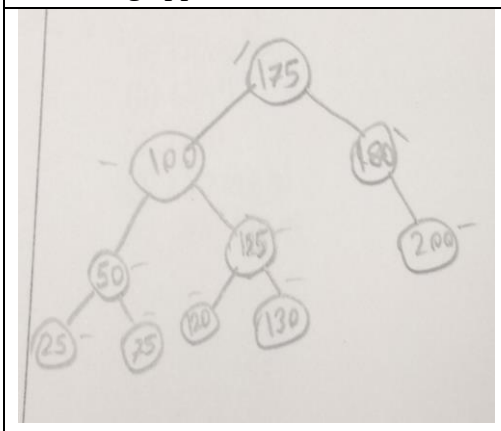
a. What keys are compared when searching for the value **127** ?

100 150 125 130 Not found!

b. Draw the tree after inserting the value **200** and rebalancing if needed. Show which case of balancing applies.



c. Draw the tree after removing the value **150** and rebalancing if needed. Identify what case of balancing applies.



Part 3: CLO3 [/ 6]

6. [3] To balance AVL Trees, it is required to store the height difference of each node in the tree. Write a recursive method **`int setHeights(AVLNode <E> p)`** as part of the AVL Tree class that sets the height difference parameter in all nodes of the AVL Tree with a root p. The method returns a integer indicating the height difference stored at p.

Hint: Follow these rules to set the height parameter in each node of the tree

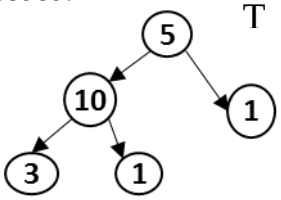
1. If **`p.left`** is null and **`p.right`** is null then **`p.height`**=0;
2. If **`p.left`** is null but **`p.right`** is not null then **`p.height`** is **`p.right.height+1`**
3. If **`p.right`** is null but **`p.left`** is not null then **`p.height`** is **`p.left.height+1`**
4. If both child nodes exist then **`p.height`** is **`Math.abs(p.left.height-p.right.height)`**

```
public int setHeights(AVLNode p)
    {
        if(p.right==null && p.left==null)
            return 0;
        else
        {
            if(p.right==null &&p.left!=null)
                p.height=setHeights(p.left)+1;
            else if(p.right!=null &&p.left==null)
                p.height=setHeights(p.right)+1;
            else
                p.height=(int)Math.abs(setHeights(p.left)-setHeights(p.right));
            return p.heights;
         }
}
```

7.[3] An Arabic version of the in-order traversal method for a binary tree is needed. In this version, the order of visiting the nodes is as follows:

1. Visit the right subtree.
2. Visit the vertex (root of current subtree).
3. Visit the left subtree.

T



For example, the Arabic inorder traversal for tree T would give 1, 5, 1, 10, 3. Give a method PrintInorderA(BTNode<E> p) to print the nodes of the tree in Arabic inorder traversal form.

```
public void printInOrder(BTNode p)
        {
            if(p==null)
                System.out.println("");
            else
                System.out.println(printInOrder(p.right) + p.value
                + printInOrder(p.left));
        }
```